IN THE SPECIFICATION

Please amend the paragraph beginning on page 4, line 11, as follows:

~~FIG. 2 shows~~ FIGs. 2A-2B show a second example spreadsheet with formulae embedded in some of the spreadsheet cells.

Please amend the paragraph beginning on page 4, line 13, as follows:

FIG. 3 shows the example spreadsheet of ~~FIG. 2~~ FIGs. 2A-2B with the formulae and several cells hidden.

Please amend the paragraph beginning on page 4, line 15, as follows:

~~FIG. 4 shows~~ FIGs. 4A-4B show a cell relation graph for the spreadsheet of ~~FIG. 2.~~ FIGs. 2A-2B.

Please amend the paragraph beginning on page 4, line 16, as follows:

FIG. 5 shows the example spreadsheet of ~~FIG. 2~~ FIGs. 2A-2B in a state of partial construction.

Please amend the paragraph beginning on page 5, line 28, as follows:

FIGs. ~~2~~ 2A-2B and 3 show how a user could construct a graphical clock in Forms/3. Consider clock 200's thirteen cells shown in ~~FIG. 2~~ FIGs. 2A-2B with their formulas, including two input cells (upper left) (cells 205 and ~~206)~~ 206 in FIG. 2A) that could eventually be replaced with references to the system clock, one output cell (middle left) (cell 210 and formula 210a), and several cells used in intermediate calculations (right) (cells 215-224 and formulae ~~251a-224a).~~ 251a-224a in FIG. 2B). The term *input cell* refers to cells whose formulas contain only constants. The term output cell refers to cells not referenced in any formula. After the programming is finished, the cells that calculate intermediate results can be hidden, and other cells rearranged, to reach the user view shown in FIG. 3.

Please amend the paragraph beginning on page 8, line 4, as follows:

Test adequacy criteria provide help in selecting test data and in deciding whether a program has been tested "enough." Test adequacy criteria have been well researched for imperative languages (e.g., (Clarke89, Frankl88, Ntafos84, Perry90)), where they are often defined on abstract models of programs rather than on code itself. An abstract model for spreadsheet languages is created, called a *cell relation graph* (CRG). A CRG is a pair (V, E),

where V is a set of *formula graphs*, and E is a set of directed edges connecting pairs of elements in set V. ~~FIG. 4 depicts~~ FIGs. 4A-4B depict the CRG for Clock 200. ~~FIG. 4 shows~~ FIGs. 4A-4B show the interrelation of each formula graph 205b, 206b, 210b, and 215b-224b: which formula graphs depend on which other formula graphs.

Please amend the paragraph beginning on page 8, line 13, as follows:

Each formula graph in set V models flow of control within a cell's formula, and is comparable to a control flow graph representing a procedure in an imperative program (Aho86, Rapps85). There is one formula graph for each cell in the spreadsheet. The process of translating an abstract syntax tree representation of an expression into its control flow graph representation is well known (Aho86); a similar translation applied to the abstract syntax tree for each formula in a spreadsheet yields that formula's formula graph. For example, ~~FIG. 4 shows~~ FIGs. 4A-4B show the formula graphs for each of the cells in Clock 200, delimited by dotted rectangles. In the formula graphs, nodes labeled "E" and "X" are *entry* and *exit* nodes, respectively, and represent initiation and termination of the evaluation of formulas. Nodes with multiple out-edges (represented as rectangles) are *predicate* nodes. Other nodes are *computation* nodes. Edges within formula graphs represent flow of control between expressions, and edge labels indicate the value to which conditional expressions must evaluate for particular paths to be taken. For example, formula graph 210b for theClock includes entry node 405 (numbered 55), exit node 410 (numbered 59), and nodes 415, 420, and 425 (numbered 56, 57, and 58, respectively). Formula graph 210b also includes edges 430, 435, 440, 445, and 450.

Please amend the paragraph beginning on page 9, line 26, as follows:

The set E of edges in the CRG, *cell dependence edges*, models dependencies between cells. ~~FIG. 4 depicts~~ FIGs. 4A-4B depict these edges by dashed lines. Each edge encodes the fact that the destination cell refers to the input cell in its formula; thus, the arrows show direction of dataflow. Note that cell dependence information is typically available to evaluation engines within spreadsheet systems as a consequence of the need to evaluate formulas; thus, this information need not be specially calculated for use in CRGs.

Please amend the paragraph beginning on page 15, line 15, as follows:

Algorithm `CollectAssoc` of Table 2 is triggered when a new formula is added, to collect new static du-associations. FIG. 6 shows a flowchart of algorithm `CollectAssoc`. Lines 2-5 (steps 605 and 610) collect du-associations involving uses in *C*. Lines 6-9 (steps 615 and 620) collect du-associations involving definitions (of *C*) in *C*. For example, referring back to FIG. 5, suppose that the most recent formula entered is that for cell `minuteY` 515. Note that its value is displayed, even though the spreadsheet has not been completely entered; when the evaluation engine is triggered to display this value, it collects *C.DirectConsumers*, *C.DirectProducers*, *C.LocalDefs*, and *C.LocalUses* for `minuteY` 515 (as done previously for the other cells on display when their formulas were entered). Called with cell `minuteY` 515, `CollectAssoc` employs this information to collect six new du-associations, described using the node numbers of ~~FIG. 4~~ FIGs. 4A-4B as: (2, (19, 20), `minute`), (2, (19, 21), `minute`), (2, 20, `minute`), (2, 21, `minute`), (20, 50, `minuteY`), and (21, 50, `minuteY`).

Please amend the paragraph beginning on page 17, line 19, as follows:

To track execution traces, which in Task 3 will enable the incremental computation of du-associations that have been exercised, it is sufficient to insert a probe into the evaluation engine. When cell *C* executes, this probe records the execution trace on *C*'s formula graph, storing it in *C.Trace*, adding only O(1) to the cost of execution. For example, in the case of `Clock`, at the moment depicted in FIG. 5, the execution trace stored for cell `minuteY` 515, described in terms of ~~FIG. 4's~~ the node ~~numbers,~~ numbers of FIGs. 4A-4B, is (18, 19, 20, 22). If the cell is subsequently reevaluated, the old execution trace is replaced with the new one. Storing only the most recent execution trace in *C.Trace* is sufficient for coverage computation because the cumulative coverage in *C.DUA* is updated incrementally during validation, as described in the discussion of Task 3.

Please amend the paragraph beginning on page 23, line 11, as follows:

To reflect the new test adequacy status of the spreadsheet whenever a cell is modified, the system must (1) update *C*'s static du-association and dynamic execution trace information, and (2) update the *exercised* flags on all du-associations affected by the modification, allowing calculation and display of new border and arrow colors to reflect the new "testedness" status of the spreadsheet. Validation tab statuses on cells that were dependent on (consumers of) *C* must also be adjusted, changing all checkmarks to question

marks if the cell retains any exercised du-associations after affected associations have been reset, or to blank if all the cell's exercised flags are now unset. (Because all of *C*'s du-associations have been reset, the validation tab for *C* is changed to blank.) For example, in the completed `Clock` spreadsheet of ~~FIG. 2,~~ FIGs. 2A-2B, if the user changes cell `minutex`'s 215 formula, then the du-associations involving `minutex` 215, and the validation statuses for `minutex` 215, `minuteHand` 222, and `theClock` 210 must all be adjusted to blank. On the other hand, if the user changes the value of `minute` 205, the validation statuses for cells `minutex` 215, `minutey` 216, `hourx` 217, `houry` 218, `hourWithFraction` 219, `fraction` 220, `hourHand` 221, `minuteHand` 222, and `theClock` 210 are adjusted to question marks or to blanks, depending on the cells' other du-associations' exercised statuses.